Optimized Localization of Underwater Wireless Sensors Using Differential Evolution Algorithm

Md. Rifat Hossain

ID: 2015-1-60-191

Nabila Bilkis Shova

ID: 2015-1-60-039

Md. Anower Hossain

ID: 2015-1-60-036

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

EAST WEST UNIVERSITY

Dhaka-1212, Bangladesh

April 2019



DECLARATION

I hereby declare that, this project has been done by us under the supervision of **Dr. Anisur Rahman, Assistant Professor, Department of Computer Science and Engineering, East West University**. I also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma. Any material reproduced in this project has been properly acknowledged.

Countersigned Signature

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**Dr. Anisur Rahman Md. Rifat Hossain**

**Supervisor Id: 2015-1-60-191**

Signature

. . . . . . . . . . . . . . . . . .. ..

**Nabila Bilkis Shova**

**Id: 2015-1-60-039**

Signature

. . . . . . . . . . . . . . . . . .. ..

**Md. Anower Hossain**

**Id: 2015-1-60-036**

ABSTRACT

Localization technology has been a core component for Wireless Sensor Network (WSN). This paper delimitates a new method of determining the co-ordinates of underwater deployed sensors. If there is no information about the initial location, we have to consider about global localization. We have used Differential Evolution algorithm (DE) in this regard, that has gained a decent reputation for solving optimization problems efficiently. As positions of the submerged sensors play a vital role for the significance of the validity of data, determining the co-ordinates precisely becomes very crucial. Previously, Cayley-Menger determinant and Multilateration technique have been used in this field. According to our proposed method, the deployed sensor can be localized with the help of only one beacon node. We have tried to make this model applicable in real life orientation and modified the algorithm accordingly with our own objective function for this specific problem domain where the beacon node has no prioriknowledge about the location of the deployed sensors. This paper aims at achieving as much localization accuracy of the submerged sensors as possible.

ACKNOWLEDGEMENT

We would like to express our gratitude to our parents. Their diligent support and perpetual inspiration towards study since the early stage of our education, has placed us at the edge of our graduation degree. I believe that whatever we have achieved and whatever we are going to gain are owing to our parents.

We would like to pay homage to our supervisor Dr. Anisur Rahman, His cordial directions have kept us on the right track from the very first day of supervision. Whenever, we came up with complicated issues, he guided us the simple way to resolve the issues. Besides, we are grateful to all our course directors for providing us with contemporary insights from the field of system development and implementation.

Our special thanks to all our friends, colleagues for their continuous inspiration and guidelines throughout my study period in East West University.

Moreover, we heartily thank to my family members for their financial supports for our study. Without their support, our study in this university could have been a dream, nothing more. We are profoundly grateful to our Creator that we have been in touch with and guided by such great individuals in the world.

Table of Content

[DECLARATION i](#_Toc5368076)

[ABSTRACT ii](#_Toc5368077)

[ACKNOWLEDGEMENT iii](#_Toc5368078)

[Table of Content iv](#_Toc5368079)

[List of Figures vi](#_Toc5368080)

[List of Tables vii](#_Toc5368081)

[Chapter 1 1](#_Toc5368082)

[Introduction 1](#_Toc5368083)

[1.1 Overview 1](#_Toc5368084)

[1.2 Motivation 1](#_Toc5368085)

[1.2.1 Why Differential Evolution 1](#_Toc5368086)

[1.3 Objectives 2](#_Toc5368087)

[1.4 Contribution 2](#_Toc5368088)

[Chapter 2 3](#_Toc5368089)

[Background Study 3](#_Toc5368090)

[2.1 Differential Evolution 3](#_Toc5368091)

[2.2 Control Parameters of DE 4](#_Toc5368092)

[2.2.1 Mutation Factor F 4](#_Toc5368093)

[2.2.2 Crossover Rate Cr 4](#_Toc5368094)

[2.2.3 Population Size NP 5](#_Toc5368095)

[2.3 Optimization Using DE 5](#_Toc5368096)

[2.4 Working Mechanism of DE 6](#_Toc5368097)

[2.4.1 Algorithm 1: The procedure of DE 6](#_Toc5368098)

[2.5 Related Works 9](#_Toc5368099)

[Chapter 3 11](#_Toc5368100)

[Proposed Model 11](#_Toc5368101)

[3.1 Overview 11](#_Toc5368102)

[3.2 Problem Domain 11](#_Toc5368103)

[3.2.1 Proposed Method 12](#_Toc5368104)

[3.2.2 Environmental Constraints 12](#_Toc5368105)

[3.3 Implementation of DE 13](#_Toc5368106)

[Chapter 4 16](#_Toc5368107)

[Simulation and Result Analysis 16](#_Toc5368108)

[4.1 Simulation 16](#_Toc5368109)

[4.2 Result Analysis 22](#_Toc5368110)

[Chapter 5 24](#_Toc5368111)

[Conclusion and Future Work 24](#_Toc5368112)

[5.1 Conclusion 24](#_Toc5368113)

[5.2 Future Work 24](#_Toc5368114)

[Bibliography 25](#_Toc5368115)

[Appendix A 28](#_Toc5368116)

List of Figures

[Fig. 2.1: Empirical distributions of candidate trial vectors for three different Cr values. (a) Cr = 0. (b) Cr = 0.5. (c) Cr = 1.0. 5](#_Toc5367983)

[Fig. 2.2: Main stages of DE algorithm 6](#_Toc5367984)

[Fig. 3.1: Co-ordinate calculation from six different positions of beacon node 12](#_Toc5367985)

[Fig. 3.2: Flowchart of proposed model 14](#_Toc5367986)

[Fig. 4.1: Error from original position for 200 iterations 22](#_Toc5367987)

[Fig. 4.2: Error from original position for 250 iterations 23](#_Toc5367988)

[Fig. 4.3: Error from original position for 300 iterations 23](#_Toc5367989)

List of Tables

[Table 1: Result for Mutation factor 0.5 and Crossover rate 0.6 16](#_Toc5367897)

[Table 2: Result for Mutation factor 0.5 and Crossover rate 0.8 17](#_Toc5367898)

[Table 3: Result for Mutation factor 0.5 and Crossover rate 0.9 17](#_Toc5367899)

[Table 4: Result for Mutation factor 0.6 and Crossover rate 0.6 18](#_Toc5367900)

[Table 5: Result for Mutation factor 0.6 and Crossover rate 0.8 19](#_Toc5367901)

[Table 6: Result for Mutation factor 0.6 and Crossover rate 0.9 19](#_Toc5367902)

[Table 7: Result for Mutation factor 0.6 and Crossover rate 0.9 20](#_Toc5367903)

[Table 8: Result for Mutation factor 0.8 and Crossover rate 0.8 21](#_Toc5367904)

[Table 9: Result for Mutation factor 0.8 and Crossover rate 0.9 21](#_Toc5367905)

Chapter 1

# Introduction

## 1.1 Overview

In this paper, we have introduced an efficient way of underwater sensor detection with a beacon node using Differential Evolution Algorithm. For that purpose, we have analyzed previously worked methods for using certain controlling parameters that affect sensor node detection efficiently. The localization problem is formulated to be a minimizing optimization problem and Differential Evolution (DE) is applied to solve this optimization problem for obtaining the estimated location of unknown sensor node.

## 1.2 Motivation

The main motivation of this thesis is to determine a new method of finding the coordinates of submerged sensors with a single beacon and optimize the problem accordingly. Many applications use underwater sensors for measuring data. The measurements of these data are meaningless without knowing the location from where the data are obtained [1]. So, locations of the sensors need to be determined for meaningful interpretation of the sensed data [2].

### 1.2.1 Why Differential Evolution

Differential Evolution (DE) was first invented in [3] as a meta-heuristic algorithm which usually uses stochastic search technology. Compared with other meta-heuristic algorithms, DE algorithm mainly has three merits: faster convergence, fewer parameters and more robustness. Unlike traditional evolutionary algorithms, the DE-variants perturb the current generation population members with the scaled differences of randomly selected and distinct population members. Therefore, no separate probability distribution has to be used for generating the offspring. For these reasons, our concentrated problem would be benefited from the powerful nature of DE.

## 1.3 Objectives

The objective of localization algorithms is to obtain the exact positions of the submerged sensors by measuring distances between beacon and nodes. Only measurement available here to compute is the distance and typically it is considered as optimization problem where objective functions to be minimized have residuals of the distance equations. Here, our main objective is to determine the co-ordinates of the underwater sensors as accurately as possible. For this purpose, we needed to find and incorporate the appropriate objective function that would optimize the result with each iteration.

## 1.4 Contribution

Despite varieties of application of UWSN, idea of submerged wireless communication has got attention of researchers since last decades. Accurate localization in underwater is needed in a range of applications, such as estuary monitoring and pollutant tracking [1]. Underwater wireless sensor network (UWSN) is envisioned to enable application for oceanographic data collection and offshore exploration for the profusion of wealth underwater world has. Marine life helps determine the very nature of our planet at a very fundamental level and for its sustenance. It, therefore, becomes crucial to obtain accurate environmental data using underwater sensors to provide and maintain sanctuary for the marine life. In this regard, optimized localization can contribute largely in advanced robot navigation, autonomous underwater vehicle control and surveillance, finding lost objects, estuary monitoring and pollutant.

Chapter 2

# Background Study

## 2.1 Differential Evolution

Differential Evolution algorithm is a branch of evolutionary programming developed by Rainer Storn and Kenneth Price (Price and Storn, 1997) for optimization problems [5]. In DE, each variable’s value is represented by a real number. The advantages of DE are its simple structure, ease of use, speed and robustness. DE is one of the best genetic type algorithms for solving problems with the real valued variables. Differential Evolution is a design tool of great utility that is immediately accessible for practical applications. DE has been used in several science and engineering applications to discover effective solutions to nearly intractable problems without appealing to expert knowledge or complex design algorithms. If a system is amenable to being rationally evaluated, DE can provide the means for extracting the best possible performance from it. Differential Evolution (DE) and Genetic Algorithm (GA) have many similarities in implementation. The operators of GA are used in DE [4]. The foremost contrast between Genetic Algorithms and Differential Evolution is that Genetic Algorithm depend on crossover, a mechanism of probabilistic and helpful trade of data among answers for find better solutions, while evolutionary strategies use mutation as the essential search mechanism. It has been seen that in classic DE, evolution is conducted for every member of the population and the resulting individuals are saved in the memory as the next generation [5].

## 2.2 Control Parameters of DE

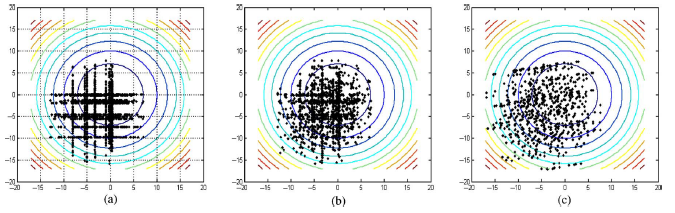
There are three main control parameters of the DE algorithm: the mutation scale factor F, the crossover constant Cr and the population size NP. A good volume of research work has been undertaken so far to improve the ultimate performance of DE by tuning its control parameters.

### 2.2.1 Mutation Factor F

Mutation is a divergence operation. Since the goal is to bring the population to convergence, mutation should happen less frequently and typically only affects a few members of a population (if any) in any given generation. A good initial choice of *F* is 0.5 [17].  It could make it worse if mutation is too high, because valuable partial solutions can be lost before convergence. The effective range of *F* is usually between 0.4 and 1.

### 2.2.2 Crossover Rate Cr

The parameter *Cr* controls how many parameters in expectation are changed in a population member. For low value of *Cr*, a small number of parameters are changed in each generation. On the other hand, high values of *Cr* (near 1) cause most of the directions of the mutant vector to be inherited prohibiting the generation of axis orthogonal steps. This effect has been illustrated in Fig. 1.1 by showing for three values of *Cr*, an empirical distribution of candidate trial vectors obtained by running DE on a single starting population of ten vectors for 200 generations with selection disabled [18].



#### Fig. 2.1: Empirical distributions of candidate trial vectors for three different Cr values. (a) Cr = 0. (b) Cr = 0.5. (c) Cr = 1.0.

### 2.2.3 Population Size NP

The effect of population size parameter on the performance of DE is not given much importance and is taken as a constant value depending on the dimension of the problem (2D to 40D). If the population size is small, the algorithm may converge fast; but the probability of premature convergence and stagnation may be higher [19]. In contrast, a large population with a strategy having good exploration capacity reduces the probability of premature convergence and stagnation effects, but the convergence speed can be slower.

## 2.3 Optimization Using DE

Scientists and engineers from all disciplines often have to deal with the classical problem of search and optimization. Optimization means the action of finding the best-suited solution of a problem within the given constraints and flexibilities. While optimizing performance of a system, we aim at finding out such a set of values of the system parameters for which the overall performance of the system will be the best under some given conditions. DE utilizes *NP* variables as population of *D* dimensional parameter vectors for each generation. The initial population is chosen randomly if no information is available about the problem. In the case of the available preliminary solution, the initial population is often generated by adding normally distributed random deviations to the preliminary solution. The basic idea behind DE is a new scheme for generating trial parameter vectors. It generates new parameter vectors by adding the weighted difference vector between two population members to a third member. If the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector replaces the vector with which it was compared. In addition, the best parameter vector is evaluated for every generation in order to keep track of the progress that is made during the optimization process [7].

## 2.4 Working Mechanism of DE

The Differential Evolution operations are based on a natural evolution principle whose aim is to keep the population diversity. Mainly, it uses *mutation* as a search mechanism and *selection* to direct the search toward the prospective regions in the feasible region.

#### Fig. 2.2: Main stages of DE algorithm

### 2.4.1 Algorithm 1: The procedure of DE

**Input:** Population *NP*; Dimension: *D*; Maximum Generation: *Gmax*

**Output:** The best vector

01: **Initialization:** Generate *NP D*-variables individuals

02: Evaluate every individual xi

03: *Generation* = 1;

04: **While** *Generation* < *Gmax do*

05: **for** *i* = 1 to *NP* // *Mutation* and *Crossover* operator

06: Generate mutation vectorvigusing Eq. (2) //DE/rand/1

07: **for** *j* = 1 to *D*

08: Generate trial vector uigusing Eq. (8)

09: **end**

10: Evaluate trial vector uig

11: **if** *fit* (uig ) < *fit* (xig )**then**

12: xig+1 = uig

13: **else**

14: xig+1 = xig

15: **end**

16: **end**

17: *Generation* = *Generation*+1;

18: **end**

19: Compare all fitness values to get the best individual

20: **Return** the best individual

**1. Initialization:**

For a given search space, *NP* individuals are generated by the initialization method and every individual is a *D* dimensions vector. Each individual in *g*th generation is denoted as Xgi = (xgi,1, xgi,2, … xgi,D), where i=1, 2,…,NP, g=0,1,2,…Gmax and Gmax is the maximum generations. In addition, Xmin= (xmin,1, xmin,2,…, xmin,D) represents the lower search bound and Xmax = (xmax,1, xmax,2,…, xmax,D) represents the upper search bound. The initial value of the *i*th individual is generated by the following method,

x0i,j = xmin,j + rand(0,1) \* (xmax,j - xmin,j), j = 1, 2, …, D (1)

Where rand (0,1) is a random number in the range 0 to 1.

**2. Mutation:**

For each individual in the population, it may be selected for mutation operation. At this time, the selected individual is the target vector, and the generated vector is the donor vector. Usually, we use the symbol “DE/*a*/*b*” to mark mutation operators. In this symbol, “*a*” stands for the basic vector and “*b*” represents the number of difference vectors that are utilized. The following six equations are most widely used in mutation operator.

*1)”DE/rand/1”*

vig = xr1g + F \* (xr2g – xr3g) (2)

*2)”DE/rand/2”*

vig = xr1g + F \* (xr2g – xr3g) + F \* (xr4g – xr5g)   (3)

*3)”DE/best/1”*

vig = xbestg + F \* (xr1g – xr2g) (4)

*4)”DE/best/2”*

vig = xbestg + F \* (xr1g – xr2g) + F \* (xr3g – xr4g) (5)

*5)”DE/current-to-best/1”*

vig = xig + F \* (xbestg – xig) + F \* (xr1g – xr2g)  (6)

*6)”DE/current-to-rand/1”*

vig = xig + K \* (xr1g – xig) + F \* (xr2g – xr3g)(7)

xbestg is the best individual in the current population. *r*1*, r*2*, r*3*, r*4*, r*5 are random integer from 1 to *NP* and they are not equal to each other. The parameter *F* is called scaling factor and *F* € [0,1].

**3. Crossover:**

The target vector xig and donor vector vigobtained in the mutation operationare used to generate trial vector ui,jg. The process of crossover is as follows,

(8)

Where *CR* is the crossover rate, *CR* € [0,1], and jrandis a random integer in the range 1 to D.

**4) Selection:**

The *selection* operator is a very greedy operation which employs a one-to-one selection strategy as follows,

(9)

The pseudo-code of DE is shown in Algorithm-1.

## 2.5 Related Works

A good number of works have been put in efforts regarding Underwater Wireless Sensor Network (UWSN). Mostly, Multilateration technique has been used to determine the location of the submerged sensors with respect to three or more known beacon nodes where distance between them is measured considering the roundtrip time of acoustic signal. However, this method of measuring distances gives erroneous results due to a number of factors, including relative angular stand of the nodes. In [9], 3D Euclidean distance estimation method requires the need of a certain number of neighboring nodes to measure inter-node distances and where error is propagated through the system due to its recursive nature. In [10], the authors have proposed a localization scheme based on buoys moored to the waterbed and mobile nodes that need to communicate directly with these buoys to get their location. This method does not support dynamic environment because buoys need to be deployed in advance in known locations. In [11], four different positions are used to obtain the beacon nodes positions of a 3D local positioning system (LPS). In [12], Duff and Muller proposed a method to solve the multilateration equations by means of nonlinear least square optimization when positions are not known. The algorithm is based on degree-of-freedom analysis, which says enough measurements from different positions will provide enough equations to solve the problem. In [13], same technique is used incorporating extended Kalman filter. However, the degree-of-freedom analysis does not guarantee a unique solution in a system on nonlinear equations, such as trilateration, when the only data available is the distance measured between the nodes [14]. In [15], Guevara et al. introduced a new closed-form solution where no position information of nodes is required to determine the positions of multiple static beacon nodes, the only information they used is the distance measurement between static beacons and mobile node.

A mathematical model is represented in [16] to determine the coordinates of submerged sensors using a single beacon for both parallel state and non-parallel state scenarios. The model has better immunity from multipath fading and linearization process of non-linear equations, resulting in more precise location of the sensors.In [8], a novel high accurate localization algorithm is proposed based on DV-Hop and DE called DECHDV-Hop for WSN. In order to reduce the further localization error, DE algorithm is used to locate unknown nodes by formulating the location estimation process as an optimization problem. The proposed algorithm DECHDV-Hop provides more accurate localization without requiring additional hardware.

Chapter 3

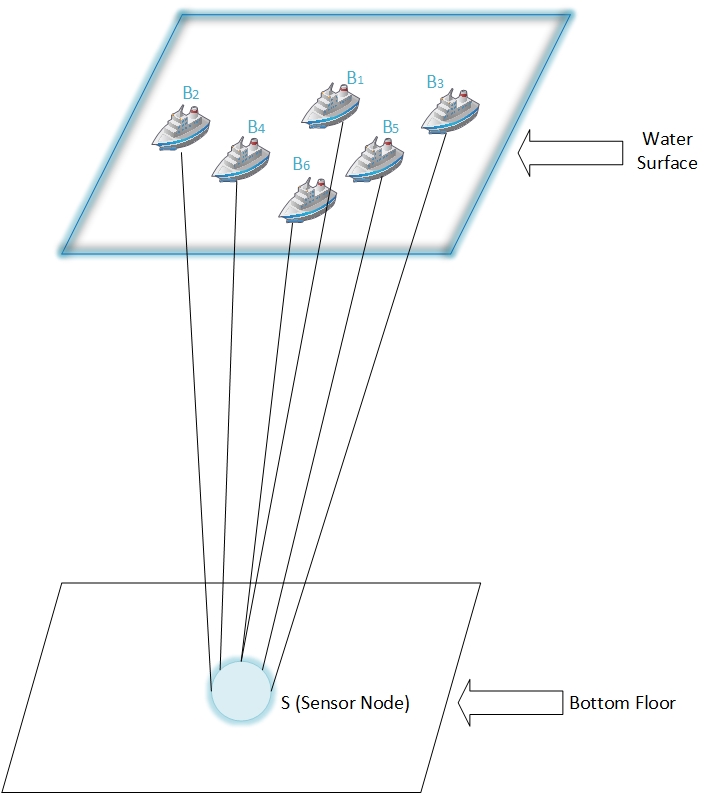
# Proposed Model

## 3.1 Overview

In the previous chapters, we have discussed the importance of determining the accurate location of the submerged sensors and optimizing the results in this regard. Having analyzed the various studies discussed in the previous chapter, in this chapter we have talked about the procedures we have followed to implement the Differential Evolution algorithm in order to determine the co-ordinates of the submerged sensors.

## 3.2 Problem Domain

We have to determine the location of a submerged sensor with the help of a beacon node. For an unknown sensor node, we will apply DE algorithm to estimate its location after getting the estimated distances. In this regard, the estimated distances are measured from 6 different positions of the beacon. To get distance measurement from six different positions of the beacon, it has been moved around in six different co-ordinates randomly within close proximity. All the distances have been measured with incorporated errors due to Gaussian noise, water temperature and other atmospheric circumstances.



#### Fig. 3.1: Co-ordinate calculation from six different positions of beacon node

### 3.2.1 Proposed Method

We are trying to localize a submerged sensor S. We have a beacon node B and we will get the co-ordinates of the sensor node in respect with that beacon node. For better coverage, we have taken distance measurements from six different positions of beacon node. It is to be noted that, measurement from six different positions is not mandatory as it was for Cayley-Menger’s determinant [16]. Any more than two positions of beacon node would give optimized result in our proposed method.

### 3.2.2 Environmental Constraints

Normally, underwater environment is more adverse than terrestrial environment; despite those limitations, it poses some merits that could be exploited in determining coordinates. Water body is relatively more homogeneous because the usual obstacles present in water are smaller in size than that of in terrestrial environment. The region of interest on the ground is more likely occupied with buildings and trees which are the major factors for multipath propagation.

Regarding signal propagation in water, acoustic signal propagates much further compare to radio signal; however, the speed of the acoustic signal is much slower than that of radio signal. The main environmental variable that we assume in our method to determine distances is the speed of acoustic signals in water. It depends on the temperature, salinity and permeability. How the speed of acoustic will vary because of aforesaid factors is not considered in this study, but assuming the presence of the constraints, we have incorporated some errors along with the measured distances so that it can be applicable in real world circumstances.

## 3.3 Implementation of DE

Before applying DE algorithm to estimate the location of an unknown node, first we have to give the definition of the objective function for DE.

(10)

Where M is the population size and Nk is the number of beacon node positions. is the estimated distance from beacon to sensor node and , and are the co-ordinates of the sensor node. , and are the co-ordinates of the beacon node.

This is a minimized optimization problem that means the smaller the function value of individual is, the closer the actual location of unknown node is. The steps we have followed in our proposed method are given below.

#### Fig. 3.2: Flowchart of proposed model

It is worth to note that in this method we have implemented, each sensor node's location estimation process is independent to that of others. In other words, to locate multiple unknown nodes, each unknown node needs an independent DE optimization process to estimate its location. Each independent optimization process includes the following processes.

The specific implementation of DE to estimate location is as follows.

**(1) Initialization:**

Initialize parameters of DE, including population size *NP*, the scaling factor *F* and the crossover rate *CR.* For an unknown node, generate *NP* individuals with two variablesaccording to Eq. (1). The form of each individual is as follows, which represents the coordinatesof a node. Then calculate the fitness of all individuals in the population based on Eq. (10).

**(2) Mutation:**

Generate adonor vector by applying “DE/best/1” (Eq. (4)) for each individual in the population. If the elements of the donor vector violate the bound constraints, it will be repaired by setting its values as its closest boundary value.

**(3) Crossover:**

Generate a trial vector by employing the binomial crossover operator (Eq. (8)) for each individual in the population. In this paper, we have used a greater *CR* value for higher convergence. Moreover, the sensitivity of *CR* will be analyzed experimentally.

**(4) Selection:**

Calculate the fitness of each trial vector based on Eq. (10), selecting the better one between the trial vector (offspring) and its corresponding target vector (parent) to go into the next generation.

The above processes of Mutation, Crossover, and Selectionkeep looping until the stop condition is met. The final optimal solution is the estimated location of the unknown sensor node. It is worth re-emphasizing that each unknown node needs an independent DE optimization process to estimate its location.

Chapter 4

# Simulation and Result Analysis

## 4.1 Simulation

In order to validate our proposed method, it has been simulated using Matlab [22]. A sensor node is placed at (80, 40, 50) and the mobile beacon moved randomly in a plane. Here, analysis has been done for 200, 250 and 300 iterations. Population size is taken 50 in all the cases. Comparison for different values of mutation factor and crossover rate is showed in the tables below.

##### **Table 1:** Result for Mutation factor 0.5 and Crossover rate 0.6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **F = 0.5, C = 0.6** | | | | | | |
|
| X | Y | Z | Error Rate | Average | | |
| X | Y | Z |
| 200 | 85.34 | 42.72 | 41.52 | 0.02 | 82.96 | 41.59 | 44.08 |
| 83.33 | 40.71 | 48.64 | 0.06 |
| 82.62 | 40.45 | 46.09 | 0.03 |
| 78.93 | 44.12 | 41.49 | 0.14 |
| 84.60 | 39.93 | 42.67 | 0.07 |
| 250 | 82.90 | 41.24 | 48.66 | 0.02 | 83.31 | 41.22 | 47.07 |
| 83.07 | 41.16 | 44.24 | 0.14 |
| 83.64 | 41.21 | 45.81 | 0.04 |
| 83.12 | 40.72 | 48.96 | 0.06 |
| 83.83 | 41.77 | 47.68 | 0.12 |
| 300 | 83.61 | 40.33 | 47.92 | 0.08 | 83.20 | 40.1 | 47.70 |
| 82.81 | 39.69 | 48.78 | 0.02 |
| 83.05 | 39.5 | 46.52 | 0.13 |
| 82.65 | 40.62 | 48.55 | 0.05 |
| 83.91 | 40.33 | 46.73 | 0.05 |

##### **Table 2:** Result for Mutation factor 0.5 and Crossover rate 0.8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **F = 0.5, C = 0.8** | | | | | | |
|
| x | Y | Z | Error Rate | Average | | |
| X | Y | Z |
| 200 | 84.17 | 40.92 | 47.39 | 0.06 | 83.58 | 40.74 | 47.49 |
| 83.67 | 40.01 | 45.71 | 0.02 |
| 83.01 | 40.49 | 48.17 | 0.04 |
| 84.14 | 41.79 | 47.89 | 0.09 |
| 82.90 | 40.49 | 48.31 | 0.09 |
| 250 | 83.49 | 40.59 | 49.24 | 0.05 | 82.91 | 40.39 | 48.95 |
| 83.47 | 40.78 | 49.74 | 0.02 |
| 83.20 | 40.09 | 48.52 | 0.03 |
| 81.51 | 39.78 | 47.51 | 0.01 |
| 82.91 | 40.74 | 49.75 | 0.05 |
| 300 | 82.71 | 39.88 | 48.75 | 0.08 | 83.34 | 40.68 | 48.87 |
| 82.73 | 40.90 | 48.57 | 0.04 |
| 83.07 | 40.84 | 48.99 | 0.04 |
| 83.57 | 41.26 | 48.36 | 0.05 |
| 84.59 | 40.54 | 49.68 | 0.12 |

##### **Table 3:** Result for Mutation factor 0.5 and Crossover rate 0.9

| **Iteration** | **F = 0.5, C = 0.9** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|
| x | Y | Z | Error Rate | Average | | |
| X | Y | Z |
| 200 | 83.74 | 40.31 | 49.03 | 0.02 | 83.35 | 40.69 | 48.41 |
| 84.04 | 40.59 | 48.29 | 0.04 |
| 83.34 | 41.14 | 46.95 | 0.09 |
| 82.91 | 40.47 | 48.49 | 0.06 |
| 82.70 | 40.93 | 49.27 | 0.04 |
| 250 | 82.16 | 41.77 | 49.39 | 0.11 | 83.22 | 40.83 | 49.39 |
| 83.38 | 40.77 | 49.49 | 0.02 |
| 83.29 | 40.52 | 49.78 | 0.05 |
| 83.67 | 40.77 | 49.10 | 0.09 |
| 83.60 | 40.35 | 49.20 | 0.06 |
| 300 | 83.35 | 40.14 | 49.82 | 0.07 | 82.70 | 40.11 | 49.18 |
| 82.06 | 38.63 | 47.52 | 0.02 |
| 82.13 | 40.58 | 49.49 | 0.05 |
| 83.04 | 40.88 | 49.15 | 0.07 |
| 82.92 | 40.33 | 49.89 | 0.08 |

##### **Table 4:** Result for Mutation factor 0.6 and Crossover rate 0.6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **F = 0.6, C = 0.6** | | | | | | |
|
| x | Y | Z | Error Rate | Average | | |
| X | Y | Z |
| 200 | 79.85 | 38.22 | 38.62 | 0.08 | 82.97 | 41.04 | 41.94 |
| 82.96 | 40.54 | 44.92 | 0.06 |
| 85.71 | 39.81 | 40.18 | 0.01 |
| 84.01 | 41.42 | 44.06 | 0.06 |
| 82.31 | 45.21 | 41.90 | 0.10 |
| 250 | 84.29 | 36.40 | 31.01 | 0.10 | 83.73 | 40.89 | 40.33 |
| 85.65 | 43.02 | 38.22 | 0.06 |
| 84.82 | 43.16 | 36.78 | 0.05 |
| 82.63 | 41.71 | 49.10 | 0.06 |
| 81.24 | 40.13 | 46.51 | 0.07 |
| 300 | 85.88 | 40.65 | 42.90 | 0.11 | 83.77 | 40.89 | 45.57 |
| 83.52 | 42.10 | 46.04 | 0.04 |
| 84.51 | 40.39 | 44.99 | 0.07 |
| 83.49 | 40.19 | 44.95 | 0.04 |
| 81.45 | 41.14 | 48.94 | 0.03 |

##### **Table 5:** Result for Mutation factor 0.6 and Crossover rate 0.8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **F = 0.6, C = 0.8** | | | | | | |
|
| x | Y | Z | Error Rate | Average | | |
| X | Y | Z |
| 200 | 83.56 | 41.56 | 42.93 | 0.07 | 82.86 | 40.58 | 47.06 |
| 82.35 | 41.53 | 49.79 | 0.01 |
| 83.29 | 40.53 | 49.25 | 0.07 |
| 82.01 | 38.61 | 44.90 | 0.02 |
| 83.11 | 40.68 | 48.45 | 0.05 |
| 250 | 84.01 | 41.74 | 47.63 | 0.03 | 83.52 | 41.19 | 47.67 |
| 83.49 | 41.18 | 46.64 | 0.05 |
| 83.79 | 40.63 | 47.13 | 0.12 |
| 83.98 | 40.78 | 48.49 | 0.04 |
| 82.35 | 41.64 | 48.47 | 0.05 |
| 300 | 82.83 | 40.60 | 49.43 | 0.08 | 82.73 | 40.59 | 49.68 |
| 82.16 | 40.45 | 49.83 | 0.05 |
| 83.06 | 40.10 | 49.94 | 0.07 |
| 83.04 | 40.93 | 49.71 | 0.03 |
| 82.58 | 40.88 | 49.48 | 0.06 |

##### **Table 6:** Result for Mutation factor 0.6 and Crossover rate 0.9

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **F = 0.6, C = 0.9** | | | | | | |
|
| x | Y | Z | Error Rate | Average | | |
| X | Y | Z |
| 200 | 81.98 | 41.20 | 47.52 | 0.04 | 82.14 | 40.85 | 48.24 |
| 83.46 | 39.59 | 46.92 | 0.04 |
| 81.62 | 41.72 | 49.83 | 0.06 |
| 83.36 | 39.75 | 48.38 | 0.04 |
| 80.31 | 42.01 | 48.57 | 0.08 |
| 250 | 82.13 | 40.88 | 47.99 | 0.10 | 82.45 | 40.75 | 48.90 |
| 82.46 | 40.89 | 49.51 | 0.05 |
| 82.94 | 40.53 | 48.53 | 0.06 |
| 82.66 | 40.96 | 49.58 | 0.06 |
| 82.04 | 40.50 | 48.87 | 0.05 |
| 300 | 81.20 | 41.09 | 48.89 | 0.03 | 82.53 | 40.85 | 49.60 |
| 84.79 | 41.68 | 48.48 | 0.03 |
| 82.93 | 40.85 | 49.45 | 0.02 |
| 82.67 | 40.47 | 49.49 | 0.04 |
| 81.09 | 40.15 | 48.69 | 0.03 |

##### **Table 7:** Result for Mutation factor 0.6 and Crossover rate 0.9

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **F = 0.8, C = 0.6** | | | | | | |
|
| x | Y | Z | Error Rate | Average | | |
| X | Y | Z |
| 200 | 86.52 | 40.12 | 36.92 | 0.12 | 84.42 | 41.40 | 38.66 |
| 84.32 | 41.00 | 34.65 | 0.06 |
| 85.47 | 40.93 | 44.22 | 0.06 |
| 84.32 | 37.97 | 38.42 | 0.04 |
| 81.44 | 46.98 | 39.07 | 0.08 |
| 250 | 85.29 | 40.04 | 42.52 | 0.05 | 84.32 | 41.33 | 41.97 |
| 83.03 | 41.55 | 41.80 | 0.06 |
| 84.31 | 41.36 | 41.86 | 0.06 |
| 83.61 | 44.25 | 43.43 | 0.04 |
| 85.34 | 39.46 | 40.24 | 0.03 |
| 300 | 85.27 | 43.77 | 38.93 | 0.04 | 82.05 | 42.31 | 42.49 |
| 82.38 | 41.24 | 46.22 | 0.07 |
| 79.99 | 44.97 | 40.12 | 0.04 |
| 78.45 | 40.06 | 45.37 | 0.06 |
| 84.17 | 41.49 | 41.84 | 0.05 |

##### **Table 8:** Result for Mutation factor 0.8 and Crossover rate 0.8

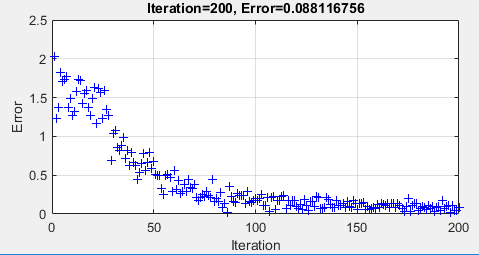
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **F = 0.8, C = 0.8** | | | | | | |
|
| x | Y | Z | Error Rate | Average | | |
| X | Y | Z |
| 200 | 77.34 | 46.57 | 42.71 | 0.06 | 80.96 | 42.90 | 42.44 |
| 78.02 | 42.34 | 41.41 | 0.08 |
| 85.31 | 41.13 | 43.57 | 0.10 |
| 80.07 | 42.57 | 39.57 | 0.05 |
| 84.06 | 41.89 | 44.96 | 0.07 |
| 250 | 83.14 | 39.71 | 46.03 | 0.06 | 82.96 | 39.85 | 43.98 |
| 82.73 | 35.97 | 36.94 | 0.08 |
| 81.65 | 40.54 | 42.00 | 0.06 |
| 84.39 | 42.32 | 48.24 | 0.05 |
| 82.89 | 40.70 | 46.67 | 0.04 |
| 300 | 82.06 | 39.97 | 48.03 | 0.04 | 82.42 | 40.47 | 47.21 |
| 82.47 | 40.14 | 46.02 | 0.08 |
| 81.71 | 40.37 | 49.50 | 0.05 |
| 83.32 | 40.26 | 43.17 | 0.08 |
| 82.54 | 41.60 | 49.30 | 0.01 |

##### **Table 9:** Result for Mutation factor 0.8 and Crossover rate 0.9

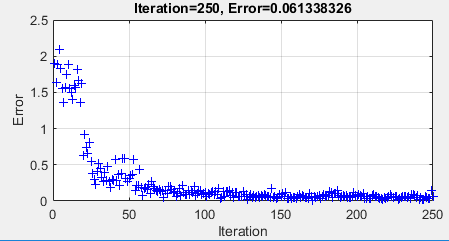
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **F = 0.8, C = 0.9** | | | | | | |
|
| x | Y | Z | Error Rate | Average | | |
| X | Y | Z |
| 200 | 82.94 | 41.12 | 47.72 | 0.08 | 82.92 | 40.71 | 47.58 |
| 82.90 | 41.52 | 46.46 | 0.05 |
| 82.79 | 39.46 | 45.63 | 0.05 |
| 84.22 | 41.01 | 48.95 | 0.04 |
| 81.72 | 40.45 | 49.16 | 0.08 |
| 250 | 82.20 | 40.28 | 48.40 | 0.05 | 82.97 | 41.04 | 47.99 |
| 83.86 | 42.40 | 48.69 | 0.07 |
| 83.17 | 40.74 | 46.36 | 0.10 |
| 82.56 | 40.84 | 48.88 | 0.02 |
| 83.03 | 40.96 | 47.64 | 0.03 |
| 300 | 83.75 | 40.19 | 48.79 | 0.04 | 82.84 | 40.44 | 48.95 |
| 83.87 | 39.98 | 46.89 | 0.02 |
| 82.48 | 40.78 | 49.69 | 0.04 |
| 81.99 | 40.86 | 49.70 | 0.08 |
| 82.13 | 40.37 | 49.68 | 0.04 |

## 4.2 Result Analysis

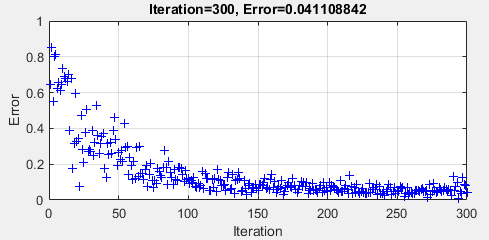
Analyzing the results, it can be said that more iterations give better result in determining the co-ordinates. We have agreed to the fact that the combination of mutation factor, F = 0.6 and crossover rate, CR = 0.9 give more accurate result than others in our proposed method. Population size has been considered constant for all the cases. The error rate is the difference of distance between the estimated value and the real position of the sensor. Results for iteration 200, 250 and 300 iterations are given below.



#### Fig. 4.1: Error from original position for 200 iterations



#### Fig. 4.2: Error from original position for 250 iterations



#### Fig. 4.3: Error from original position for 300 iterations

Chapter 5

# Conclusion and Future Work

## 5.1 Conclusion

In this paper, we have discussed a method for determining the co-ordinates of a submerged sensor. We have used Differential Evolution algorithm as it is one of the best algorithms for optimization problems. In this method, the co-ordinates of the sensor are calculated with respect to the position of a beacon node. For better convergence, six different positions of the beacon are considered. Results from simulation validate the proposed method that generates negligible error in co-ordinates determination of the sensors when distances between beacon and sensors are true Euclidean. It further shows that coordinates are within acceptable error range when Gaussian noise is applied to distance determination as we have incorporated some errors in distance measurements for real life orientation. The result gets more accurate with greater number of iteration.

## 5.2 Future Work

In future study, we have plans to implement our method using only one position of the beacon node. We also plan to consider involuntary mobility of the submerged sensor due to currents in the proposed method. Real life implementation can be done to validate the method in applicable situations. The main constraint in this method we have implemented, each sensor node's location estimation process is independent to that of others. So, to locate multiple unknown nodes, each unknown node needs an independent DE optimization process to estimate its location. In near future, we will try to locate multiple sensors with only one process of DE.

Bibliography

[1] J. H. Cui, J. Kong, M. Gerla, and S. Zhou, "The challenges of building mobile underwater wireless networks for aquatic applications," *Network, IEEE,* vol. 20, pp. 12-18, 2006.

[2] H. P. Tan, R. Diamant, W. K. G. Seah, and M. Waldmeyer, "A survey of techniques and challenges in underwater localization," Ocean Engineering, vol. 38, pp. 1663-1676, 2011.

[3] J. Kennedy R. Eberhart, Particle swarm optimization, Neural Networks, 1995. IEEE International Conference on Neural Networks, 4 (1995) 1942-1948.

[4] S. Zhao, S. Chai and B. Zhang, "Dynamic differential evolution strategy localization for Wireless Sensor Networks in three-dimensional space," *2016 35th Chinese Control Conference (CCC)*, Chengdu,2016, pp. 8423-8427.

[5] Q. Wei and X. Qiu, "Dynamic Differential Evolution algorithm with composite strategies and parameter values self-adaption," *2015 Seventh International Conference on Advanced Computational Intelligence (ICACI)*, Wuyi,2015, pp. 271-274.

[6] Storn, R., Price, K.: Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)

[7] V. ARUNACHALAM, "Optimization Using Differential Evolution", 2009.

[8] L. Cui, C. Xu, G. Li, Z. Ming, Y. Feng and N. Lu, "A high accurate localization algorithm with DV-Hop and differential evolution for wireless sensor network", *Applied Soft Computing*, vol. 68, pp. 39-52, 2018.

[9] Z. Zhou, J. H. Cui and S. Zhou, "Efficient localization for large-scale underwater sensor networks," Ad Hoc Networks, vol. 8, pp. 267-279,2010.

[10] T. C. Austin, R. P. Stokey, and K. M. Sharp, "PARADIGM: a buoybased system for AUV navigation and tracking," in OCEANS 2000 MTS/IEEE Conference and Exhibition, 2000, pp. 935-938 vol.2.

[11] J. C. Prieto, A. R. Jiménez, J. Guevara, J. L. Ealo, F. Seco, J. O. Roa, and F. Ramos, "Performance evaluation of 3D-LOCUS advanced acoustic LPS," Instrumentation and Measurement, IEEE Transactions on, vol. 58, pp. 2385-2395, 2009.

[12] P. Duff and H. Muller, "Autocalibration algorithm for ultrasonic location systems," in Wearable Computers,2003. Proceedings. Seventh IEEE International Symposium on, 2003, pp. 62-68.

[13] E. Olson, J. Leonard, and S. Teller, "Robust range-only beacon localization," in Autonomous Underwater Vehicles, 2004 IEEE/OES, 2004, pp. 66-75.

[14] J. Guevara, A. R. Jimenez, A. S. Morse, J. Fang, J. C. Prieto, and F. Seco, "Auto-localization in Local Positioning Systems: A closedform range-only solution," in Industrial Electronics (ISIE), 2010 IEEE International Symposium on, 2010, pp. 2834-2840.

[15] J. Guevara, A. Jiménez, J. Prieto, and F. Seco, "Auto-localization algorithm for local positioning systems," Ad Hoc Networks,2012.

[16] A. Rahman and V. Muthukkumarasamy, "Localization of Submerged Sensors with a Single Beacon for Non-Parallel Planes State", in *ICUFN*, Prague, Czech Republic, 2018.

[17] R. Storn and K. V. Price, “Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces,” ICSI, USA, Tech. Rep. TR-95-012, 1995 [Online]. Available: http://icsi.berkeley.edu/∼storn/litera.html

[18] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the art", *IEEE Trans. on Evolutionary Computation,* Vol. 15, No. 1, pp. 4-31, Feb. 2011.

[19] R. Mallipeddi and P. N. Suganthan, "Empirical study on the effect of population size on Differential evolution Algorithm," *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, Hong Kong,2008, pp. 3663-3670.

[20] M. N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization. IEEE T. Evolut. Comput. 18 (3) (2014) 378-393.

[21] L. Cui, G. Li, Z. Zhu, Z. Wen, N. Lu, J. Lu, A novel differential evolution algorithm with a self-adaptation parameter control method by differential evolution. Soft Comput. 3 (2017) 1-20.

[22] Differential Evolution Homepage. Retrieved from http://www1.icsi.berkeley.edu/~storn/code.html#matl

Appendix A

# Codes

## Main Function

%Initialization

N = 3; % Number of variables

M = 50; % Populations size 50

F = 0.6; % Mutation factor

C = 0.9; % Crossover rate

I\_max = 200; % Max iteration time

Run = 1; % Number of test time

X\_max = [150, 150, 50];

X\_min = [-150, -150, 0];

Func=@OBJ; %Objective function

%Whole test loop

for r=1:Run

iter=0;

%Generate MxN matrix

for m=1:M

for n=1:N

X(m,n) = X\_min(n) + rand() \* (X\_max(n)-X\_min(n));

end

end

for i=1:I\_max % Stop when the iteration is larger than the max iteration time

iter=iter+1;

for m=1:M % For each individual of the population

% Mutation

[V]=rand1(X,M,F,m);

% Check if any element in the V matrix is beyond the boundary

for n=1:N

if V(1,n)>X\_max(1,n)

V(1,n)=X\_max(1,n);

end

if V(1,n)<X\_min(1,n)

V(1,n)=X\_min(1,n);

end

end

% Crossover

jrand=floor(rand()\*N+1);

for n=1:N

R1=rand();

if (R1<C || n==jrand)

U(1,n)=V(1,n); %put the result in U matrix

else

U(1,n)=X(m,n);

end

end

% Selection

if Func(U(1,:)) < Func(X(m,:))

Tr=U(1,:);%U(1,:);

else

Tr=X(m,:);

end

% Use the selection result to replace the m row

X(m,:)=Tr;

% Evaluate each individual's fitness value, and put the result in the Y matrix

Y(m,:)=Func(X(m,:));

Z(m,:)=X(m,:);

end % Now the 1th individual generated

% Select the lowest fitness value

[y,ind1]=sort(Y,1);

Y\_min=y(1,:);

[Ymin,ind] = min(Y);

fprintf('Least distance difference corresponding to 6 beacons: ');

disp(Ymin);

z = mean(Ymin);

fprintf('fittest co-ordinates: ');

disp(Z(ind,:));

W=mean(Z);

fprintf('Most fit co-ordinate is: ');

disp(W);

fprintf('Error Rate is: ');

disp(z);

% plot the picture of iteration

figure(1);

plot(iter,z,'r+');

xlabel('Iteration');

ylabel('Error');

title(sprintf('Iteration=%d, Error=%9.9f',i,z));

grid on;

hold on;

end % Finish I\_max times iteration

end

## Objective Function

function y = OBJ (X)

% Beacon Nodes

B1 = [0 0 0];

B2 = [-100 -120 0];

B3 = [-80 130 0];

B4 = [140 -70 0];

B5 = [60 120 0];

B6 = [-90 -130 0];

% Sensor Node

S = [80 40 50];

%Euclidean distance from beacon nodes to sensor node

d1 = pdist2(B1,S,'euclidean');

d2 = pdist2(B2,S,'euclidean');

d3 = pdist2(B3,S,'euclidean');

d4 = pdist2(B4,S,'euclidean');

d5 = pdist2(B5,S,'euclidean');

d6 = pdist2(B6,S,'euclidean');

% Adding Gaussian Error

%1st beacon

for i=1:10

d1 = d1 + rand();%erf(d1);

d11(i)=d1;

end

davg1 = mean(d11);

%2nd beacon

for i=1:10

d2 = d2 + rand();%erf(d2);

d12(i)=d2;

end

davg2 = mean(d12);

%3rd beacon

for i=1:10

d3 = d3 + rand();%erf(d3);

d13(i)=d3;

end

davg3 = mean(d13);

%4th beacon

for i=1:10

d4 = d4 + rand();%erf(d4);

d14(i)=d4;

end

davg4 = mean(d14);

%5th beacon

for i=1:10

d5 = d5 + rand();%erf(d5);

d15(i)=d5;

end

davg5 = mean(d15);

%6th beacon

for i=1:10

d6 = d6 + rand();%erf(d6);

d16(i)=d6;

end

davg6 = mean(d16);

%distance from beacon nodes to DE generated sensor nodes

D1 = pdist2(B1, X,'euclidean');

D2 = pdist2(B2, X,'euclidean');

D3 = pdist2(B3, X,'euclidean');

D4 = pdist2(B4, X,'euclidean');

D5 = pdist2(B5, X,'euclidean');

D6 = pdist2(B6, X,'euclidean');

y = [abs(D1-davg1), abs(D2-davg2), abs(D3-davg3), abs(D4-davg4), abs(D5-davg5), abs(D6-davg6)];

end